White Paper Report

Report ID: 98501

Application Number: HD5060609

Project Director: Douglas Reside (dreside@umd.edu)

Institution: University of Maryland, College Park

Reporting Period: 4/1/2009-3/31/2011

Report Due: 6/30/2011

Date Submitted: 7/31/2011

When MITH proposed to build the Collaborative Ajax Modelling Platform (CAMP), a tool for collaboratively building 3-D models in a web browser, we knew we were undertaking a difficult endeavor. In 2008 when we applied for the grant, Google's Chrome web browser had just been released and Internet Explorer 6 was still one of the most common ways users interacted with the web. When we wrote the application, we proposed to use Scalar Vector Graphics XML to create basic 3-D graphics that could be managed with a back-end tool like Subversion. It was an ambitious goal, but one we felt would be particularly appropriate for an NEH Startup grant designed to encourage risky experimentation and innovation.

Almost immediately after receiving the grant, Google released the Google O3D toolkit which promised to deliver cross-browser compatible 3-D graphics in the web. A few truly stunning demo pages and a few early successes using the toolkit convinced our team that CAMP should shift from SVG XML to Google O3D. We built a couple of quick prototypes in O3D, but found the package difficult to use and poorly documented. The help forums, while initially populated with excited users, became more and more sparse until, about 9 months into the project, Google pulled the plug on O3D altogether.

Forced to rethink the project again, we went back to our initial use case: a tool for a faculty member in theater architecture history to create collaboratively edited models of historic theaters. These theaters were rooted in a historical place and time, and so we decided to try to build on the existing tools in Google Earth to create COLLADA models for presentation in the 3-D geographic visualization tool. Unfortunately, by this time MITH did not have any excess capacity in its own staff and the remaining funds were insufficient to hire a new developer, so we decided to contract the work to an independent programmer who had produced many successful digital humanities projects in the past. The contractor was most comfortable working in Ruby and the Rails framework rather than in MITH's usual PHP environment, but as we did have an installation of Ruby running on our servers and a desire to complete the work as efficiently as possible, we agreed that the new version of CAMP could be designed in the developers preferred language.

Unfortunately, although the code worked on local machines, we found that it could not be made to work in the server environment we had available at the University of Maryland. Differing versions of the Ruby interpreter and the permissions required by several of the Ruby Gems proved more difficult to reconcile with our server environment than we had anticipated. There is much promise in this code base, and we make it available for download and reuse at http://peach.umd.edu/svn/CAMP, but ultimately we wanted to deliver a version of CAMP that could be installed on our (admittedly unusual) server environment.

In the final months of the grant, one of MITH's interns, Jon Gilmour, had developed an alternative, Adobe Flash-based interface for CAMP using ActionScript and the open source Flex SDK. The program was capable of exporting models into the COLLADA format which could be imported into Google SketchUp and Google Earth with relative ease. We were so impressed with the ease of use of the tool and with its indifference to server environment that as we continued to struggle with the next steps for the project, adapting this prototype into the official version of CAMP became more and more attractive. MITH's R & D developer Travis Brown, added a backend connection from Gilmour's interface to the popular open source versioning repository, GitHub, to manage collaboration. The result, the fourth iteration of CAMP, is itself available in GitHub and on MITH's website at mith.umd.edu/camp/demo and https://github.com/umd-mith/Flex4Modeler

Lessons Learned

We are proud of the end product of CAMP, but we feel one of it's most useful contributions to digital humanities may be in the lessons we learned and can now share about Digital Humanities software development at the edge of the possible:

1. **Be willing to fail quickly:** We do not regret shifting from SVG to Google O3D, but it was probably a mistake to hold as doggedly to O3D as we did even after it was clear that most developers and even Google were abandoning the platform. There is always a chance that things will get better, but if no significant progress has been made after a month of hard work, changing directions is advised.
2. **Require contractors to use your own environment:** Even when things are desperate and good will abounds, it's smart to insist that those you hire to work in the software environment you know you can support. MITH did not have the capacity to sustain a Ruby application even if it had been successfully launched on our servers. We wanted results quickly, but it would have been smarter to make choices that would have made life easier for ourselves in the long or medium term.
3. **Devote sufficient time to evaluating and selecting frameworks and libraries:** The second iteration of CAMP used the gdalwarp utility from the Geospatial Data Abstraction Library (GDAL: http://www.gdal.org/) to perform georectification of theater floorplans and maps from Google. This functionality can be computationally expensive, however, and the approach taken to integrating the library with the Ruby on Rails application proved not to be sufficiently robust, resulting in frequent server crashes and runaway processes. A more careful matching of framework and library could potentially have made this integration easier. This iteration of CAMP also used an open source library providing OpenID authentication for Ruby on Rails applications that we discovered is not sufficiently well documented or supported, and that has been difficult to extend and maintain.
4. **Allow staff to experiment with alternative options:** We weren't sure we would ever use Jon Gilmour's Flash code, and, in fact, imagined his work on the tool would be more an educational experience for him than of any real benefit to us. Having an alternative to our earlier plans, though, proved exceptionally beneficial. There is much fretting in the digital humanities about reinventing the wheel, but our experience demonstrates that occasionally parallel and seemingly duplicative streams of development can produce very good results.